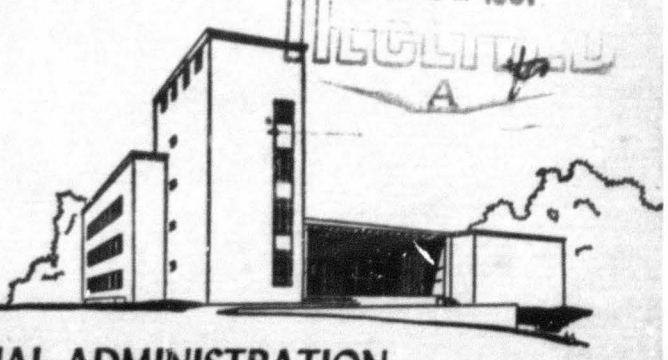
Carnegie Institute of Technology

Pittsburgh 13, Pennsylvania



GRADUATE SCHOOL of INDUSTRIAL ADMINISTRATION

William Larimer Mellon, Founder

Management Sciences Research No. 103

ON SOME SEQUENCING PROBLEMS[1/]

by

Wlodzimierz Szwarc*

June, 1967

1/ From Przeglad Statystyczny Vol. IX No. 4 (1962) pp. 367-382 and Vol. X No. 1 (1963) pp. 139-154. Translated by the author with editorial assistance and suggestions from W. W. Cooper for the Management Sciences Research Group. Acknowledgment is gratefully made to Professor Cooper for his many helpful suggestions and comments.

* Technical University, Wroclaw, Poland and International Center of Operations Research and Econometrics (CORE) University of Louvain, Belgium.

MANAGEMENT SCIENCES RESEARCH GROUP
GRADUATE SCHOOL OF INDUSTRIAL ADMINISTRATION
CARNEGIE INSTITUTE OF TECHNOLOGY
PITTSBURGH, PENNSYLVANIA 15213

1. Introduction:

Consider the following problem, which is one of the classical sequencing problems: We are to produce on m machines one single aggregate consisting of n different items each of which is to be operated on by some or by all of the machines. The order of processing each item through the machines is given. We are also given the operation time for each item on each machine. It is assumed that at any moment:

1^{o}. No machine is able to handle more than one item

2^{o}. No item can be operated on by more than one machine.

Given the operating time for each item on each machine the problem is to find a production program which will be called optimal program for producing the aggregate in a minimal time.

This problem was considered by S. B. Akers and I. Friedman in [1] where they presented a solution method for the mx2 case. For the general mxn case (m-machines, n-items) these authors gave a criterion which enables to check whether a program is feasible or non feasible. To solve the mxn case one must first consider all programs. Then, applying the Akers-Friedman rule, all non feasible ones are removed and then finally the optimal solution is found by examining each of the remaining programs.

This method is laborious, even for moderate values of m and n, although it should also be noted that its authors supply additional advice on how to remove non optimal plans for the mx2 case. Even so the number of remaining feasible programs to consider is still large. By means of a graphical approach, presented in [2], Akers provides an approximate method of solving the mx2 case. This was subsequently elaborated in [9] where, using Akers' graphical approach, I was able to solve the mx2 problem and

in this same reference I also supplied an approximate method for solving the mxn case.[1/] R. Bellman [3] and S. M. Johnson [7] independently solved the 2xn case on the assumption that the same order of processing the items through two machines, say A and B, is used. On this assumption Johnson also solved the 3xn problem for two special cases. L. G. Mitten [8] solved a generalization of the Bellman -Johnson 2xn problem while maintaining assumption 1^{o}, as above, plus the condition in which there are upper limits on the length of time for each item from the moment it starts on machine A until it is finished on machine B.

All of the methods mentioned above for solving the sequencing problem are combinatorial in nature.

After 1958 when R. E. Gomory [6] first published a method for solving the integer programming problem, papers appeared which treated this sequencing problem as a special case of an integer programming problem. These cases, however, involved introducing a considerable number of variables and constraints. For instance, E. H. Bowman in [4], solving a 4x3 problem by the integer programming method, deals with at least 300 variables and even more constraints. Several authors (e.g., G. B. Dantzig [5], H. M. Wagner [10]) then tried to find an integer programming formulation of this problem which would require a smallest possible amount of variables and constraints.

This paper presents solutions to the following problems.

1. A generalization of the 2xn Bellman-Johnson problem where the processing order of the items is not the same.
2. The 3xn Bellman-Johnson problem for several new special cases.
3. The 2xn Bellman-Johnson case where each item already operated on by machine A must wait until it starts on machine B.

[1/] This method may sometimes not work at all in the sense that it may lead to a program which is unfeasible.

In the first sections we present (without proof) the methods of solution to the following problems.

1. 2xn and 3xn Bellman-Johnson case (Johnson's method)

2. The mx2 Akers-Friedman problem (solved by the author of this paper)

The final section of the paper then presents cases for which the method for solving the mxn case as given in [9] is an exact method of solution.

Remark: The same symbols appearing in different sections of this paper may have different meanings.

2. Solution of the 2xn and 3xn Bellman-Johnson Case.

We illustrate the solution method by an example: There are two machines, A and B, and five items, which we denote by the numbers 1 through 5. Each of the items is first operated on by machine A and then by machine B. The operating times are given in table 1.

Table 1

	A_i	B_i	E_i
1	3	4	-1
2	5	2	3
3	4	1	3
4	6	4	2
5	2	5	-3

Hence, for instance, $B_3=1$ means that the operating time for item 3 on machine B is equal to one unit of time. The numbers E_i in the last column are equal to A_i-B_i.

Each working (and also each optimal) program is determined when we know in which order the items pass each machine. Johnson showed that to find an optimal program (also in the 3xn case) one may consider only programs where the processing order for both machines is the same. However, there exist optimal programs which do not possess this property.

The problem reduces to that of finding a processing sequence -- a permutation of n numbers $1,\ldots,n$ -- corresponding to the optimal program.

Divide the set $(1,\ldots,n)$ into two disjoint subsets s and s', where

$$s = \{(i) \mid E_i < 0\} \quad \text{and} \quad s' = \{(i) \mid E_i \geq 0\}$$

Let s and s' be sets of ℓ and $n-\ell$ elements respectively $(0 \leq \ell \leq n)$. The method of constructing the optimal sequence is as follows: Order the elements $1,\ldots,n$ so that in the sequence

a) the elements of s appear before the elements of s'

b) the corresponding numbers A_i form a nondecreasing sequence for $i$es and a nonincreasing sequence for $i$es$'$.

In our example $s = (1,5)$, $s' = (2,3,4)$. Using the rules a and b which were just given, it is easy to establish the optimal sequence, which is $(5,1,4,2,3)$. Figure 1 presents the optimal operation program corresponding to the sequence $(5,1,4,2,3)$.

Remark: The numbers shown in Figure 1 denote items

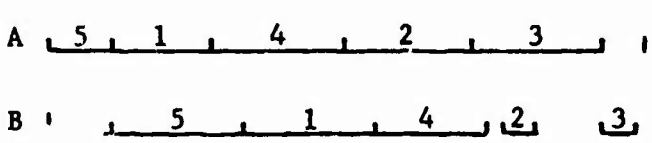


Figure 1

The minimal operating time necessary to produce the aggregate is equal to 21 units $(\sum_{i=1}^{5} A_i + 1$, or $\sum_{i=1}^{5} B_i + 5)$, where the numerical values 1 and 5 denote idle time for machines A and B respectively.

In [7] Johnson also solved the 3xn case where the processing order for all the items is A, B, C, and where for all $i,j=1,2,\ldots,n$: or $B_i \leq A_j$ or $B_i \leq C_j$.

Consider a 3x7 example with the following table of operation times.

Table 2

	A_i	B_i	C_i	A_i+B_i	B_i+C_i
1	6	2	3	8	5
2	7	2	4	9	6
3	9	6	7	15	13
4	8	6	2	14	8
5	6	4	3	10	7
6	10	2	5	12	7
7	9	1	2	10	3

As can be seen, here have the case $B_i \leq A_j$. One now finds the optimal sequence by applying the method from the 2xn case for two fictitious machines, M and N, with operating times obtained from the expression $M_i=A_i+B_i$, $N_i=B_i+C_i$. In this example s is an empty set (all M_i-N_i are positive), so $s' = (1,2,\ldots,7)$. Ordering the numbers N_i in a nondecreasing sequence we get two optimal sequences -- viz., (3,4,5,6,2,1,7) and (3,4,6,5,2,1,7).

3 Solution of the mx2 and mxn Akers-Friedman Problem.

We again illustrate the solution method by means of a 5x2 example. The passing order as well as the operation times are as follows:

Item 1 -- $A^5B^2C^4D^1E^3$

Item 2 -- $A^4C^2E^1D^2B^2$

Consider an x0y coordinate system where the x-axis coresponds to Item 1 and the y-axis to Item 2. Construct a rectangle PMQN where

$$PN = A_1+B_1+C_1+D_1+E_1 = 5+2+4+1+3 = 15$$

and

$$PM = A_2+B_2+C_2+D_2+E_2 = 4+2+2+2+1 = 11$$

Assign to machines the following "vertical" areas in ABCDE order

$0 \leq x \leq 5$	to machine	A
$5 \leq x \leq 7$	to	B
$7 \leq x \leq 11$	to	C
$11 \leq x \leq 12$	to	D
$12 \leq x \leq 15$	to	E

Assign to machines the "horizontal" areas in a ACEDB order

$0 \leq y \leq 4$	to machine	A
$4 \leq y \leq 6$	to	C
$6 \leq y \leq 7$	to	E
$7 \leq y \leq 9$	to	D
$9 \leq y \leq 11$	to	B

To each machine there corresponds a rectangle as determined by the intersection of the horizontal and vertical intervals that were associated with this machine. E.g., to machine C there corresponds the rectangle: $7 \leq x \leq 11$, $4 \leq x \leq 6$.

One may now describe each program by a continuous line with the following properties:

1. Points P and Q are on this line
2. The line does not cross any of the rectangles A,B,C,D,E.
3. The line consists of straight line segments which are either parallel to one of the axes or else forming a 45^{o} angle with the x-axis.

The total length of all vertical (horizontal) segments is the total waiting time for Item 1 (2). The total length of the projections on either axis of the 45^{o} segments is the total time when both items are operated on simultaneously (but on different machines).

The problem reduces to that of finding a line with minimal total length for the vertical segments. Alternatively, however, one may look for a line where the total length of the horizontal segments is minimal or, instead, one may search for a line with maximal total length for the 45^{o} segments. All of these problems are equivalent.

Let us introduce the following definitions and notations. By a node we mean a north-west and south-east corner of each rectangle and also points P and Q. Consider two nodes $w_1 = (x_1, y_1)$ and $w_2 = (x_2, y_2)$ such that $x_1 \leq x_2$ and $y_1 \leq y_2$ (so w_1 cannot be on the right of or above w_2). We say that node w_1 is neighboring to w_2 if one can link these nodes by a line with the properties 2 and 3 specified above.

Let w_1 be a neighboring node to w_2. We define a distance, $d(w_1 w_2)$ as follows.

$$d(w_1 w_2) = \max [0, (y_2 - y_1) - (x_2 - x_1)]$$

Let $\pi(w)$ be the set of nodes such that w is neighboring to each element of $\pi(w)$. The nodes of $\pi(w)$ lie in a rectangle whose corners are points w and Q.

Consider the set of all lines which have properties 2 and 3 linking w and Q. By a length of each such line we mean the total length of all of its vertical segments. In this set there exists a line of minimal length. Let $f(w)$ be the length of this line. Then the following is true

$$f(w) = \min_{\bar{w} \in \pi(w)} [d(w,\bar{w}) + f(\bar{w})]. \qquad (2)$$

We arrange the nodes so that their x coordinates form a nonincreasing sequence. Nodes which have the same x-coordinate we arrange in such a way that their y-coordinates form a decreasing sequence. E.g., in the example we are using we get the following sequence

(15,11), (15,6), (12,7), (11,9), (11,4), (7,9), 7,6), (5,11), (5,0), (0,4), (0,0).

The arranged nodes are denoted by $w_1,\ldots,w_k$ (w_1=Q, w_k=P). Applying (2), find the values of $f(w_s)$ for $s=2,\ldots,k$ ($f(w_1)=0$) and draw the lines of the length $f(w_s)$. Write the numbers for $f(w_s)$ above the corresponding nodes w_s. The line with length $f(w_k)=f(P)$ (in our example this is $f(w_{11})$) is the solution of the problem. This line is indicated on Figure 3 by arrows; its length is equal to 3 units. This means that the total operation time equals $A_1+B_1+C_1+D_1+E_1+3 = 18$ units. From the optimal line it is easy to read the optimal program which is presented in Table 3.

From the optimal program one can also read the optimal operation sequences for each machine, which may be written as follows: $A_{(1,2)}$, $B_{(1,2)}$, $C_{(1,2)}$, $D_{(1,2)}$, $E_{(2,1)}$.

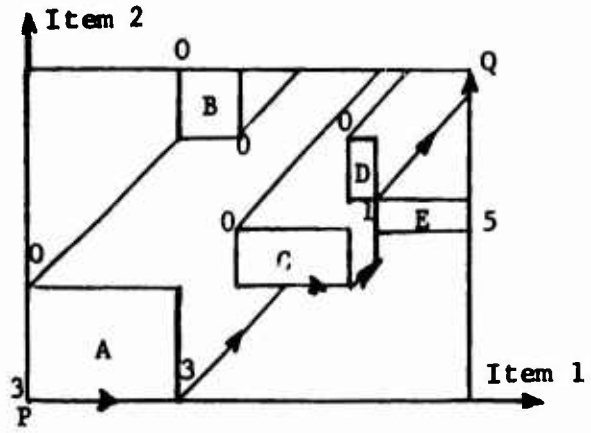


Figure 3

Symbol $C_{(1,2)}$ means that machine C will operate first item 1 and then item 2. By a program we will mean also the set of operation sequences for all machines.

Table 3

Period	Item 1 is operated on by machine	Item 2 is operated on by machine
0—5	A	-
5—7	B	A
7—9	C	A
9—11	C	-
11—12	D	C
12—13	-	C
13—14	-	E
14—16	E	D
16—17	E	B
17—18	-	B

Suppose the coordinate axes correspond to items i and j. If the line (presenting an operation program) is running between some rectangle, say F, and an axis corresponding to item i then the corresponding program will contain a symbol $F_{(i,j)}$ -- which means that machine F will operate first on item i and then on item j.

In [9] an approximate solution method is given (illustrated by a 3 x 10 example) of the mxn problem. With this method one must first solve all $\binom{n}{2}$ possible mx2 problems by the method shown in this section. Given the $\binom{n}{2}$ optimal programs, then, considering each machine separately, one constructs a program for the mxn problem (this is not always possible) and then presents it graphically in the manner shown in Figure 1.

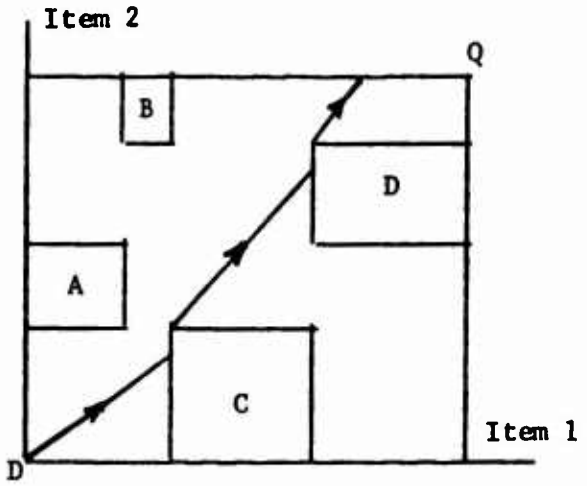


Figure 4

Let us illustrate this procedure by a 4 x 3 example, which was mentioned in Section 1 as given by Bowman in [4]. The processing order and operation times are as follows:

Item 1 -- $A^5B^2C^8D^7$

Item 2 -- $C^8A^4D^5B^3$

Item 3 -- D^6A^7.

Solving 3 problems -- one for Items 1,2 and machines A,B,C,D, another for Items 1,3 and machines A,D and a third for Items 2,3 and machines A,D -- by the method described in this section we get 3 optimal lines to which there correspond three programs $A_{(1,2)},B_{(1,2)},C_{(2,1)},D_{(2,1)}$ (see Figure 4); $A_{(1,3)},D_{(3,1)}$ and $A_{(2,3)},D_{(3,2)}$. From $A_{(1,2)},A_{(1,3)},A_{(2,3)}$ it follows that the operating program for A is $A_{(1,2,3)}$. The operating sequence for B and C as determined from the first of the three problems is $B_{(1,2)}$ and $C_{(2,1)}$. For machine D we get $D_{(3,2,1)}$.[1/] So we have found the program $A_{(1,2,3)},B_{(1,2)},C_{(2,1)},D_{(3,2,1)}$ for the 4x3 problem which is presented on Figure 5.

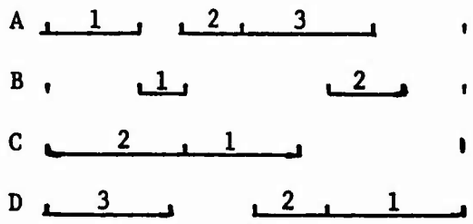


Figure 5

In general for the mxn case ($m > 2$, $n > 2$) we are not able to say whether a given program is optimal without examining all programs ($(n!)^m$ in number). For our example, however, the program we obtained is optimal. This follows from the fact that: a) the total operation time of no program for the 4x3 problem is smaller than the minimal operation time

---

[1/] Remark: It is impossible to construct a program (such a program will be called infeasible) if the solution of the second problem were $A_{(1,3)}$ $D_{(1,3)}$. Then we would deal with sequences $D_{(2,1)},D_{(1,3)},D_{(3,2)}$ from which it is impossible to derive a working plan for machine D.

of either subproblem 4x2 and 2x2. b) The total operation time for the problem is equal to the minimal operation time (24 units) of the 4x2 problem.

In [1] Akers and Friedman gave the following necessary and sufficient condition for a program of an mxn problem to be feasible: If for items $1,2,\ldots,k$ $(k \leq n)$ the processing sequences are

$$\begin{aligned}
&\text{for Item 1 -- } \ldots M^1 \ldots M^2 \ldots \\
&\text{for Item 2 -- } \ldots M^2 \ldots M^3 \ldots \\
&\qquad\vdots \\
&\text{for Item k-1 -- } \ldots M^{k-1} \ldots M^k \ldots \\
&\text{for Item k -- } \ldots M^k \ldots M^1 \ldots
\end{aligned}$$

(the machines are then said to form a k-element cycle), then the feasible program cannot be of the form

$$M^1(\ldots 2\ldots 1\ldots)\,M^2(\ldots 3\ldots 2\ldots)\cdots M^{k-1}(\ldots k\ldots k\text{-}1\ldots)\,M^k(\ldots 1\ldots k\ldots)$$

This implies that for the Bellman-Johnson case there exist no unfeasible program. This result is due to the fact that there is no cycle since the passing order is the same for all items.

4 Solution of the 2xn Case.

1. There are two machines A and B and n items where the passing order for item $1,2,\ldots,n_1$ is AB while for the remaining n_2 items, $n_1+1,\ldots,n$ $(n_1+n_2=n)$ the order is BA (for $n_1=0$ or $n_2=0$ we get the 2xn Bellman-Johnson case).

The production program is determined given A_pB_q where p as well as q are permutations of numbers $1,\ldots,n$. Let $r=(1,2,\ldots,n_1)$ and $\bar{r}=(n_1+1,\ldots,n)$. Then the Akers-Friedman feasibility theorem for the 2xn problem becomes: Program A_pB_q is feasible if and only if for any $i\in\bar{r}$ and $j\in r$ it is impossible for p and q to be simultaneously of the form

$p = (\ldots\ i\ \ldots\ j\ \ldots)$, $q = (\ldots\ j\ \ldots\ i\ \ldots)$.

Let us denote by Ω the set of feasible programs A_pB_q where the first n_1 numbers in p are elements of r while the first n_2 numbers of q belong to $\bar{r}$. The following theorem holds

Theorem 1: For each feasible program $A_p B_q$ there exists a program $A_{p*} B_{q*}$ that belongs to Ω and consumes no more time. (This will be called a "no worse" program.)

Proof: Let $A_p B_q \ddot{\epsilon} \Omega$

Then either

$p = (\ldots i,j \ldots)$ where $i \epsilon \bar{r}$, $j \epsilon r$ or
$q = (\ldots i,j \ldots)$, where $i \epsilon r$, $j \epsilon \bar{r}$.

Consider the first case; here q must be of the form $(\ldots i,\ldots,j\ldots)$ but not of the form $(\ldots j \ldots i \ldots)$ otherwise program A_pB_q would be unfeasible. Figure 6 illustrates program A_pB_q.

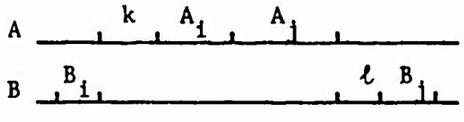


Figure 6

Consider program A_p,B_q where $p' = (\ldots j,i \ldots)$ - (p' was obtained from p by transposing i and j). This program shown on Figure 7 is feasible according to the Akers-Friedman theorem.

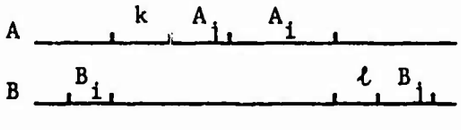


Figure 7

As seen from this figure program A_p,B_q is not worse than A_pB_q. This can be seen by observing that the time of finishing the operation of parts i and j on machine A are the same in both programs but one may finish the operations of these parts on machine B applying program A_p,B_q sooner than in program A_pB_q.

Let us turn to the second case: p must have the form $(\ldots i,\ldots j\ldots)$ where $i\epsilon r$, $j\epsilon\bar{r}$. In a way similar to the first case one may show that program A_pB_q, with $q' = (\ldots j,i,\ldots)$ is feasible and not worse than program A_pB_q.

It is easy to define a procedure leading from any feasible program A_pB_q to a feasible program $A_{p*}B_{q*}$ which belongs to Ω and is not worse than A_pB_q. This procedure will be described in an example.

Let $$A_pB_q = A_{(\bar{5},1,\bar{4},3,2)}\ B_{(\bar{5},1,\bar{4},2,3)}$$

where the dashed numbers indicate elements of $\bar{r}$ while the remaining numbers are elements of r.

For convenience write A_pB_q in the form

$$\binom{p}{q} = \begin{pmatrix} \bar{5},1,\bar{4},3,2 \\ \bar{5},1,\bar{4},2,3 \end{pmatrix}$$

Below, for instance, we show sequence of intermediate and feasible programs (each program being not worse than the preceding one) leading from

$$\binom{p}{q} \text{ to } \binom{p*}{q*} \epsilon\ \Omega$$

$$\begin{pmatrix} \bar{5},1,\bar{4},3,2 \\ \bar{5},1,\bar{4},2,3 \end{pmatrix} \rightarrow \begin{pmatrix} \bar{5},1,3,\bar{4},2 \\ \bar{5},1,\bar{4},2,3 \end{pmatrix} \rightarrow \begin{pmatrix} \bar{5},1,3,2,\bar{4} \\ \bar{5},1,\bar{4},2,3 \end{pmatrix} \rightarrow \begin{pmatrix} 1,\bar{5},3,2,\bar{4} \\ \bar{5},1,\bar{4},2,3 \end{pmatrix} \rightarrow$$

$$\rightarrow \begin{pmatrix} 1,3,\bar{5},2,\bar{4} \\ \bar{5},1,\bar{4},2,3 \end{pmatrix} \rightarrow \begin{pmatrix} 1,3,2,\bar{5},\bar{4} \\ \bar{5},1,\bar{4},2,3 \end{pmatrix} \rightarrow \begin{pmatrix} 1,3,2,\bar{5},\bar{4} \\ \bar{5},\bar{4},1,2,3 \end{pmatrix} = \binom{p*}{q*} \epsilon\ \Omega$$

Applying theorem 1 we restrict ourselves only to programs belonging to Ω (there may exist, however, optimal programs not belonging to Ω). All these programs are feasible.

2. Let $\binom{p}{q}$ be an arbitrary element of Ω. There are three cases

$$\text{Case 1:} \quad \sum_{i\in r} A_i = \sum_{i\in \bar{r}} B_i.$$

$$\text{Case 2:} \quad \sum_{i\in r} A_i > \sum_{i\in \bar{r}} B_i = w.$$

$$\text{Case 3:} \quad \sum_{i\in \bar{r}} A_i < \sum_{i\in \bar{r}} B_i.$$

Consider Case 1. Then $\binom{p}{q}$ is an optimal program since the corresponding operation time is

$$\max \left(\sum_{i\in r+\bar{r}} A_i \, , \sum_{i\in r+\bar{r}} B_i\right)^{2/}$$

and there exists no program with a smaller operation time (see Figure 8).

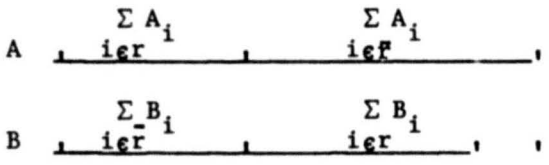


Figure 8

The number of optimal programs is equal to $(n_1!)^2\,(n_2!)^2$ which is the number of elements of the set Ω.

Consider Case 2. Program $\binom{p}{q}$ is optimal if $\sum_{i\in \bar{r}} A_i > \sum_{i\in \bar{r}} B_i$ (Case 2) for then the total operation time equals $\sum_{i\in r+\bar{r}} A_i$ and no program has a less consuming operation time. Here, like in Case 1, we also have $(n_1!)^2 \cdot (n_2!)^2$ optimal programs.

2/ The symbol $R + \bar{r}$ means a union of sets r and $\bar{r}$.

Suppose, however, that $\sum_{i\in r} A_i < \sum_{i\in r} B_i$ (remember that $\sum_{i\in r} A_i > \sum_{i\in r} B_i$). To find an optimal program one must first solve a Bellman-Johnson $2xn_1$ case where the processing order is AB but with an additional assumption (assumption w) that machine B will start at least w units later than machine A does.

Consider a $2xn_1$ case under assumption w and let $\binom{u}{v}$, $u \neq v$, be an arbitrary program for this problem. The following is true.

Theorem 2. Program $\binom{u}{u}$ or $\binom{v}{v}$ is not worse than $\binom{u}{v}$.

Proof: It is obvious (see figure 9a and 9b)

that $\binom{\ldots j,i\ldots}{\ldots j\ldots i\ldots}$ is not worse

than $\binom{\ldots i,j\ldots}{\ldots j\ldots i\ldots}$

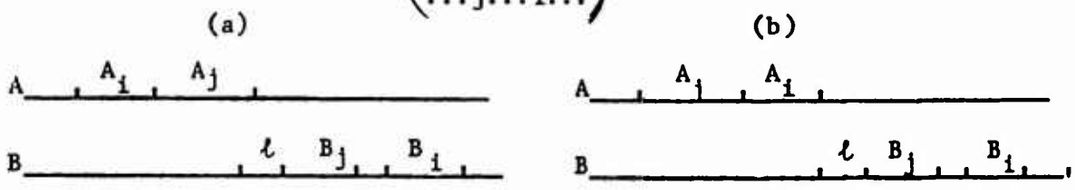


Figure 9

It is easy to develop a procedure leading from any program $\binom{u}{v}$ to a not worse one $\binom{u}{u}$ or $\binom{v}{v}$

The following example illustrates the procedure.

$$\binom{u}{v} = \binom{1,4,3,5,2}{1,2,3,4,5} \rightarrow \binom{1,3,4,5,2}{1,2,3,4,5} \rightarrow \binom{1,3,4,2,5}{1,2,3,4,5} \rightarrow \binom{1,3,2,4,5}{1,2,3,4,5} \rightarrow$$

$$\rightarrow \binom{1,2,3,4,5}{1,2,3,4,5} = \binom{v}{v}$$

This completes the proof of the theorem.

From now on we will consider only programs $\binom{p}{q}$ which are elements of Ω but such that the elements of r appear in the same order in p as well as in q. Denote the set of all such programs $\bar{\Omega}$ where $\bar{\Omega} \subset \Omega$. Now we face an $2xn_1$ Bellman-Johnson case, but with assumption w. Figure 10 shows one of the possible programs for this case.

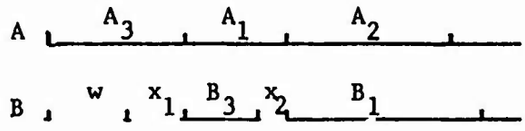


Figure 10

We are to find a permutation u of numbers $1,\ldots,n_1$ such that the operation time for program $\binom{u}{u}$ will be minimal. For such a program the total idle time for either machine will also be minimal.

By x_i let us denote the idle time (in suitable time units) of machine B after it completed item i-1 but before it starts to operate on the next item (the $i\text{-}1^{\underline{th}}$ item need not be the same as item i-1, see Figure 10).

Without loss of generality however we may assume (for convenience) that $u = (1,2,\ldots,n_1)$. Then the $i^{\underline{th}}$ item will be just item i. Also

$$x_1 = \max(A_1 - w, 0),$$

$$x_2 = \max(A_1 + A_2 - B_1 - x_1 - w, 0),$$

$$x_1 + x_2 = \max(A_1 + A_2 - B_1 - w, x_1) = \max\left(\sum_{i=1}^{2} A_i - \sum_{i=1}^{1} B_i - w, \sum_{i=1}^{1} x_i\right)$$

Si ilarly

$$\sum_{i=1}^{n_1} x_i = \max\left(\sum_{i=1}^{n_1} A_i - \sum_{i=1}^{n_1-1} B_i - w, \sum_{i=1}^{n_1-1} x_i\right) =$$

$$= \max \max_{1\le t\le n_1} (K_t - w, 0) = \max(\max_{1\le t\le n_1} K_t - w, 0),$$

where

$$K_t = \sum_{i=1}^{t} A_i - \sum_{i=1}^{t-1} B_i.$$

The problem is to find such a permutation of elements $1,2,\ldots, n_1$ for which the corresponding Σx_i is minimal. Note that $K_t \leq w$ implies that the corresponding idle time equals zero and therefore this permutation is optimal. If, however, $\max_{1\leq t\leq n_1} K_t > w$ then the total idle time for machine B is equal to $\max_{1\leq t\leq n_1} K_t - w$.

It is obvious (since w is a constant) that a permutation is optimal if $\max_{1\leq t\leq n_1} K_t$ is minimal.

Thus the problem reduces to one of finding a permutation with minimal $\max_{1\leq t\leq n_1} K_t$ and this is identical with the classical Bellman-Johnson case [7]. To solve the problem one may apply Johnson's method as given in Section 2, above. The optimal program $\binom{p}{q}$ for Case 2 has the following properties.

1) $\binom{p}{q} \in \bar{\Omega}$

2) the first n_1 elements in p and the last n_1 elements in q are arranged according to Johnson's method.

3) The order of the remaining elements in p and q may be arbitrary.

Set $\bar{\Omega}$ therefore contains $(n_2!)^2$ optimal programs.

Remark: if $\max_{v} \sum_{i\in v} x_i \leq \sum_{i\in r+\bar{r}} A_i - \sum_{i\in r+\bar{r}} B_i$, where v is an arbitrary permutation of n_1 elements of r then the solution of the problem can be the same as in Case 1.

Consider Case 3.

If in addition $\sum_{i\in r} B_i \geq \sum_{i\in \bar{r}} A_i$ then, $\binom{p}{q}$ is an optimal program since the corresponding total working time equals $\sum_{i\in r+\bar{r}} B_i$ which is minimal. Then, like in Case 2, there are $(n_1!)^2 (n_2!)^2$ optimal programs. Suppose that $\sum_{i\in r} B_i < \sum_{i\in \bar{r}} A_i = w'$. In a way similar to Case 2 we come to the following conclusion: The optimal program $\binom{p}{q}$ has the following properties.

$\binom{p}{q} \in \bar{\Omega}$

the first n_2 elements of q and the last n_2 elements of p are ordered according to Johnson's method (for the $2xn_2$ case - with machines A, B, and parts $i\in\bar{r}$)

the order of the remaining elements in p and q is arbitrary.

Remark:

if $\max \sum_{i\in v} x_i \leq \sum_{i\in r+\bar{r}} B_i - \sum_{i\in r+\bar{r}} A_i$,
then the optimal solution can be the same as in Case 1.

(v is a permutation of n_1 numbers from the set r)

Corollary: Any program $\binom{p}{q}$ posessing properties 1, 2, 3 is always optimal.

Example: Consider a 2x7 problem with $r = (1, 2, 3, 4, 5)$ and $\bar{r} = (6, 7)$. Then assume the following table of operating times obtains:

Table 4

	A_i	B_i
1	7	3
2	3	2
3	2	1
4	1	1
5	2	7
6	3	3
7	1	2

Consider an arbitrary program belonging to Ω, say program

$$\binom{p}{q} = \begin{pmatrix} 1, & 2, & 3, & 4, & 5, & 6, & 7 \\ 6, & 7, & 1, & 2, & 3, & 4, & 5 \end{pmatrix}$$

It is easy to check (by drawing a program like that in Figure 1) that its total operation time equals 22 units.

Here we have the second case because

$$\sum_{i \in r} A_i = 15 > \sum_{i \in \bar{r}} B_i = 4 \quad \text{and} \quad \sum_{i \in \bar{r}} A_i = 5 < \sum_{i \in r} B_i = 14.$$

According to the solution procedure of Case 2 we solve the Bellman-Johnson problem (machines A, B, items 1, 2, 3, 4, 5) and get two sequences 5,1,2,3,4 and 5,1,2,4,3. Therefore one can construct $2^2(n_2!)^2$ optimal programs belonging to $\bar{\Omega}$. One of those is program $\begin{pmatrix} 5, & 1, & 2, & 3, & 4, & 7, & 6 \\ 6, & 7, & 5, & 1, & 2, & 3, & 4 \end{pmatrix}$ with the total operation time equal to 19 units (see Figure 11).

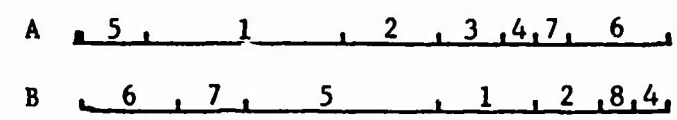


Figure 11

5 Solution of Some New Special Cases of the Bellman-Johnson Problem

There are three machines A, B, C and n items 1,2,...,n, all to be operated in order ABC. The operation times for the items are denoted, as before, by A_i, B_i, C_i while x_i, y_i will mean, respectively, the idle times for machines B_i and C_i after finishing the $i\text{-}1^{\underline{st}}$ item and before operating on the $i^{\underline{th}}$ item. Johnson proved that to find an optimal program one may restrict one's self to programs of the type $A_p B_p C_p$ where p is a permutation of n numbers $1,2,\ldots,n$. Therefore, we have to find a p such that the idle time of either machine, say machine C, will be minimal.

Without loss of generality we may assume $p = (1,2,\ldots,i,i+1,\ldots,n)$

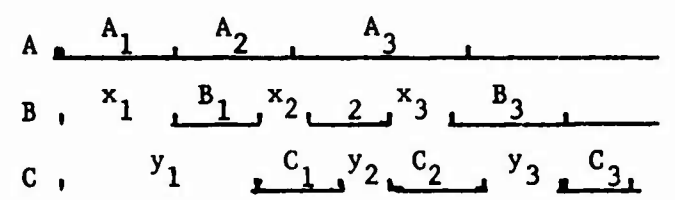


Figure 12

Then (see Figure 12)

$$y_1 = x_1 + B_1 = A_1 + B_1$$

$$y_2 = \max(x_1 + x_2 + B_1 + B_2 - y_1 - C_1, 0)$$

and for any n

$$y_n = \max\left(\sum_{i=1}^{n} x_i + \sum_{i=1}^{n} B_i - \sum_{i=1}^{n-1} y_i - \sum_{i=1}^{n-1} C_i, 0\right),$$

Therefore,

$$\sum_{i=1}^{n} y_i = \max\left(\sum_{i=1}^{n} x_i + \sum_{i=1}^{n} B_i - \sum_{i=1}^{n} C_i, \sum_{i=1}^{n-1} y_i\right).$$

Find

$$\sum_{i=1}^{n} x_i.$$

Then (see Figure 12)

$$x_i = A_1, x_2 = \max (A_1 + A_2 - x_1 - B_1, 0)$$

and

$$x_n = \max\left(\sum_{i=1}^{n} A_i - \sum_{i=1}^{n-1} x_i - \sum_{i=1}^{n-1} B_i, 0\right)$$

Therefore

$$\sum_{i=1}^{n} x_i = \max\left(\sum_{i=1}^{n} A_i - \sum_{i=1}^{n-1} B_i, \sum_{i=1}^{n-1} x_i\right).$$

By K_u denote

$$\sum_{i=1}^{u} A_i - \sum_{i=1}^{u-1} B_i, \text{ and let } X_u = \sum_{i=1}^{n} x_i.$$

Then

$$X_n = \max (K_n, K_{n-1}).$$

This implies

$$X_1 = K_1 \text{ (bo } X_0 = 0), X_2 = \max (K_2, K_1), X_3 = \max [K_3, \max(K_2, K_1)] =$$

$$= \max [K_3, K_2, K_1].$$

In general

$$X_n = \sum_{i=1}^{n} x_i = \max_{1 \le u \le n} K_u = \max_{1 \le u \le n} \left[\sum_{i=1}^{u} A_i - \sum_{i=1}^{u-1} B_i\right].$$

By H_v denote

$$\sum_{i=1}^{v} B_i - \sum_{i=1}^{v-1} C_i ;$$

Then

$$\sum_{i=1}^{n} y_i = \max(H_n + \max_{1 \le u \le n} K_u, H_{n-1} + \max_{1 \le u \le n} K_u, \ldots, H_1 + K_1) =$$

$$= \max_{1 \le u \le v \le n} (H_v + K_u) = g (p) \qquad (4)$$

Remark: This problem may be solved in a different way by fixing the end moment of the operation program (instead of fixing the starting moment) and then looking for a program with a minimal operation time, with time running in an opposite direction. The problem then reduces to one of minimizing the total idle time for machine A. Permutation $(1,2,\ldots,n)$ will then, in fact, mean permutation $(n,n-1,\ldots,1)$. The formula for the total idle time of machine A becomes

$$\max\ (\bar{H}_v + \bar{K}_u), \tag{5}$$

where

$$\bar{H}_v = \sum_{i=1}^{v} B_i - \sum_{i=1}^{v-1} A_i, \quad \bar{K}_u = \sum_{i=1}^{u} C_i - \sum_{i=1}^{u-1} B_i.$$

The graphical presentation of such a program differs from the one given in Figure 12 in that on the first row we present the working plan of machine C (which is working non stop) while the third row corresponds to machine A.

Formula (5) may be obtained in a straightforward manner from (4) by replacing A_i by C_i and vice versa.

Take a permutation $p' = (1,2,\ldots,j-1,j+1,j,j+2,\ldots,n)$ which emerges from p by transposing j and j+1. Consider

$$g(p) = \max_{1\le u\le v\le n} (H_v + K_u)\ ,\ g(p') = \max_{1\le u\le v\le n} (H'_v + K'_u).$$

It is easy to see that $K_u = K'_u$ and $H_v = H'_v$ for each u and v different from j and j+1. Examine the expressions

$$L = \max(H_j + K_u,\ 1 \le u \le j;\ \ H_{j+1} + K_u,\ 1 \le u \le j+1)$$

$$R = \max(H'_j + K'_u),\ 1 \le u \le j;\ \ H'_{j+1} + K'_u,\ 1 \le u \le j+1).$$

It is clear that if L=R then $g(p) = g(p')$ while the inequality $L < R$ implies $g(p) \le g(p')$ (the equality $g(p) = g(p')$ holds only if $H_v + K_u$ attains its maximum for u and v which differ from j as well as from j+1).

We may therefore restrict ourselves to consider L and R only. First determine numbers K'_j, K'_{j+1}, H'_j, H'_{j+1}. The following hold

$$K'_j = \sum_{i=1}^{j-1} A_i + A_{j+1} - \sum_{i=1}^{j-1} B_i = K_j - A_j + A_{j+1}.$$

$$K'_{j+1} = \sum_{i=1}^{j+1} A_i - \sum_{i=1}^{j-1} B_i - B_{j+1} = K_{j+1} + B_j - B_{j+1},$$

$$H'_j = \sum_{i=1}^{j-1} B_i + B_{j+1} - \sum_{i=1}^{j-1} C_i = H_j - B_j + B_{j+1},$$

$$H'_{j+1} = \sum_{i=1}^{j+1} B_i - \sum_{i=1}^{j-1} C_i - C_{j+1} = H_{j+1} + C_j - C_{j+1}$$

Replace in R the thus determined values of K'_j, K'_{j+1}, H'_j, H'_{j+1}. Then

$$L = \max(H_j+K_1,\ldots,H_j+K_{j-1},H_j+K_j;\ H_{j+1}+K_1,\ldots,H_{j+1}+K_{j-1},$$
$$H_{j+1}+K_j,H_{j+1}+K_{j+1})$$

$$R = \max\ (H_j-B_j+B_{j+1}+K_1,\ldots,H_j-B_j+B_{j+1}+K_{j-1},$$
$$H_j-B_j+B_{j+1}+K_j+A_j-A_{j+1}\ ,$$
$$H_{j+1}+C_j-C_{j+1}+K_1,\ldots,H_{j+1}+C_j-C_{j+1}+K_{j-1},\ H_{j+1}+C_j-C_{j+1}+$$
$$+K_j-A_j+A_{j+1},\ H_{j+1}+C_j-C_{j+1}+K_{j+1}+B_j-B_{j+1}).$$

By substracting from L and R the same value

$$\sum_{i=1}^{j+1} B_i - \sum_{i=1}^{j-1} C_i = H_{j+1} + C_j = H_j + B_{j+1}$$

we get new expressions. Call these L' and R' where

$$L' = \max(K_1-B_{j+1},\ldots,\ K_{j-1}-B_{j+1},\ K_j-B_{j+1};$$
$$K_1-C_j,\ldots,K_{j-1}-C_j,\ K_j-C_j\ ,\ K_{j+1}-C_j\ ) \qquad (6)$$

$$R' = \max(K_1 - B_j, \ldots, K_{j-1} - B_j, K_j - B_j + dA;$$

$$K_1 - C_{j+1}, \ldots, K_{j-1} - C_{j+1}, K_j - C_{j+1} + dA, K_{j+1} - C_{j+1} - dB),$$

where $dA = A_{j+1} - A_1$, $dB = B_{j+1} - B_j$.

The inequalities $L' < R'$ and $L < R$ are equivalent.

We will prove the following

Theorem 3. The optimal permutation p (the permutation p of the optimal program $A_p B_p C_p$) is to be constructed according to the following rule: Element j must appear before element j+1 -- i.e., $p = (\ldots j, \ldots j+1 \ldots)$ -- if for each $i = 1, \ldots, n$, $j = 1, \ldots, n-1$ one of the following conditions holds.

1:a) $B_j < B_{j+1}$, b) $A_j + B_j < A_{j+1} + B_{j+1}$, c) $B_j + C_j > B_{j+1} + C_{j+1}$.

2:a) $B_i > C_j$, b) $C_j > C_{j+1}$, c) $B_j + C_j > B_{j+1} + C_{j+1}$, d) $A_j - C_j < A_{j+1} - C_{j+1}$.

3:a) $B_i > A_j$, b) $A_j < A_{j+1}$, c) $A_j + B_j < A_{j+1} + B_{j+1}$, d) $A_j - C_j < A_{j+1} - C_{j+1}$

for $j = 1, 2, \ldots, n$.

Proof: Transpose two elements j and j+1 in an arbitrary permutation p of n numbers $1, \ldots, n$. Denote the new permutations by p'. Without loss of generality we may assume $p = (1, \ldots, n)$. Then $p' = (1, \ldots, j-1, j+1, j, j+2, \ldots, n)$

I. If p satisfies condition 1, then $L' < R'$, since for each $r = 1, 2, \ldots, 2j+1$ the rth element of L' is less than the rth element of R'. We will show this only for $r = j, 2j, 2j+1$. since the proof for all remaining r is obvious (see assumption 1a, 1c)

For r=j: $K_j - B_{j+1} < K_j - B_j + dA = K_j - B_j + A_{j+1} - A_j$. Hence we get

$A_j + B_j < A_{j+1} + B_{j+1}$ which holds because of 1b.

For r=2j: $K_j - C_j < K_j - C_{j+1} + dA = K_j - C_{j+1} + A_{j+1} - A_j$ which implies the relation to be proved -- viz.,

$$A_j + C_{j+1} < A_{j+1} + C_j$$

From 1c it follows that $B_j + C_j + A_{j+1} > B_{j+1} + C_{j+1} + A_{j+1}$ while 1b implies $A_{j+1} + B_{j+1} + C_{j+1} > A_j + B_j + C_{j+1}$. Applying the last two results we get

$$B_j + C_j + A_{j+1} > A_j + B_j + C_{j+1},$$

and consequently

$$A_{j+1} + C_j > A_j + C_{j+1} \quad , \quad \text{Q.e.d.}$$

For $r = 2j + 1$: $K_{j+1} - C_j < K_{j+1} - C_{j+1} - B_{j+1} + B_j$ hence

$$B_{j+1} + C_{j+1} < B_j + C_j.$$

This inequality is true in view of 1c. Thus we have proved that if p satisfies condition 1 then $L' < R'$ which in turn implies $g(p) \leq g(p')$.

The relations given by condition 1 are transitive. Consider, say, relation $A_j + B_j < A_{j+1} + B_{j+1}$. It is easy to see that for any $i,j,k = 1,\ldots,n$, $i \neq j \neq k$, the inequalities $A_i + B_i < A_j + B_j$ and $A_j + B_j < A_k + B_k$ imply

$$A_i + B_i < A_k + B_k.$$

The proof of transitivity for the other relations goes in a similar way. The property of transitivity implies the existence of a procedure for constructing a sequence of permutations starting from any given permutation and continuing to the optimal one in such a way that the corresponding numbers $g(p)$ form a nonincreasing sequence.

Let us illustrate this procedure by an example. Suppose that from 1 we have the following relations ($i \rightarrow j$ means that i precedes j)

$1 \to 2,\ 1 \to 3,\ 1 \to 4,\ 1 \to 5,\ 2 \to 3,\ 2 \to 4,\ 2 \to 5,\ 3 \to 4,\ 3 \to 5,\ 4 \to 5.$ (7)

Consider a permutation $(2,4,3,1,5) = q$. The first "disorder" in q (looking from the left side) is $4,3$ (according to (7)). Therefore permutation $\bar{p}$ is at least as good as q since $g(\bar{p}) \leq g(q)$. Removing the next "disorder" in $\bar{p}$ we get another permutation not worse than $\bar{p}$, and so on. Thus the following sequences emerge

$q = (2,4,3,1,5),\ (2,3,4,1,5),\ (2,3,1,4,5),\ (2,1,3,4,5),\ (1,2,3,4,5) = p.$

Permutation p is optimal since for any permutation p' there exists a sequence of permutations $p',\ldots,p$, where $g(p') \geq \ldots \geq g(p)$, which implies $g(p') \geq g(p)$ Q.e.d.

The rule for finding the optimal permutation under condition 1 is the following: Arrange numbers $B_j\ldots,$ in a nondecreasing sequence. The sequence of the corresponding indices is the optimal permutation.

Since we don't know, a priori, whether condition 1 holds, the following procedure is therefore proposed. Form sequences $\{B_j\}$, $\{A_j + B_j\}$ and $\{B_j + C_j\}$, the first to be nondecreasing and the last to be nonincreasing. If all three sequences which form the corresponding indices are identical, call them p, then, condition 1 holds and p is the optimal permutation.

II. We shall prove the theorem in the case where condition 2 holds (the proof of the theorem under condition 3 goes in a similar way since condition 3 can be obtained from condition 2 by replacing the symbols A_i by C_i and vice versa and by changing the direction of inequalities.).

We are to show that $L' \leq R'$.

Remark: the $r^{\underline{th}}$ term in L' for each $1 \leq r \leq j$ is not greater than the $j + r^{\underline{th}}$ term of L', the $r^{\underline{th}}$ term of R' for for $1 \leq r \leq j-1$ is less than the $j + r^{\underline{th}}$ term of L' while the j-th term of R' is less than the $2j^{\underline{th}}$ term of R'.

To prove that $L' < R'$ it is sufficient to show that the $r^{\underline{th}}$ term of L' for $j \leq r \leq 2j+1$ is less than the $r^{\underline{th}}$ term of R'. The proof will be shown only for $r = 2j, 2j+1$ since for the remaining r the proof (see condition 2b) is trivial.

For $r = 2j$: $K_j - C_j < K_j - C_{j+1} + dA$,

which leads to

$$A_j - C_j < A_{j+1} - C_{j+1}.$$

This inequality holds in view of condition 2d.

For $r = 2j+1$: $K_{j+1} - C_j < K_{j+1} - C_{j+1} - dB$ which implies

$$B_{j+1} + C_{j+1} < B_j + C_j$$

and this also holds in view 2c.

In a way similar to the earlier development, one can show that conditions 2b, 2c, 2d, are transitive which implies that p is an optimal permutation provided it satisfies condition 2. To find the optimal permutation arrange numbers $1,\ldots,n$ in such a way as to satisfy conditions 2a, 2b, 2c, 2d. If in each case we get the same sequence, say p, then p is the optimal permutation.

The following theorem holds:

Theorem 4: Let $\min\limits_{1\leq i,j\leq n} (A_i + C_j) = A_{i_o} + C_{j_o}$. If for each $i,j=1,\ldots,n$

$B_i \geq \max (A_j, C_j)$ then any permutation of the form

$(i_o,\ldots,j_o)$, $(j_o,\ldots,i_o)$ is optimal.

Proof: Denote $p = (i_o,\ldots,j_o)$, $p' = (j_o,\ldots,i_o)$. Permutation p as well as p' is optimal because the corresponding operation time for program $A_pB_pC_p$ as well as for A_p,B_p,C_p, is equal to

$$\sum_{i=1}^{n} B_i + \min_{i\neq j} (A_i + C_j)$$

and no other program has a smaller operation time.

Example. Consider a 3x5 Bellman-Johnson problem with the following table of operation times (Table 5):

	A_i	B_i	C_i
1	4	3	15
2	6	7	8
3	7	7	8
4	5	9	4
5	6	5	10

Table 5

To check whether condition 1 holds form the following table (Table 6)

	A_i+B_i	B_i	B_i+C_i
1	7	3	18
2	13	7	15
3	14	7	15
4	14	9	13
5	11	5	15

Table 6

Arranging the items in a nondecreasing sequence of B_i we get two permutations (1,5,2,3,4) and (1,5,3,2,4). Arranging the items in a nondecreasing sequence of A_i+B_i we obtain two sequences (1,5,2,3,4) and (1,5,2,4,3), while arranging B_i+C_i in a nonincreasing sequence get six permutations -- among them (1,5,2,3,4) which is the only one appearing in each case.

Therefore (1,5,2,3,4) is the optimal sequence.

6 The Sequencing Problem With a Time Lag.

The problem considered in this section is a generalization of the 2xn Bellman-Johnson case which is obtained by introducing a condition under which there must be some waiting time of at least $D_i - A_i$ duration for item i after it is finished on machine A and before it starts on machine B. (The processing order is AB for all the items). The case when, for all i, $D_i - A_i \leq 0$, reduces to a classical Bellman-Johnson problem. The problem considered by Mitten in [8] is also a generalization of the Bellman-Johnson case. The problem considered here, however, is not a special case of Mitten's problem, or vice versa. Indeed Mitten introduced time lags because he assumed that the same item can be handled by two machines simultaneously.

Remark: The Johnson procedure (as well as the notations) are similar to that in [8].

By $A_i, B_i, D_i - A_i$, denote operation times and idle times for item i. Let t_i' and t_i be the corresponding instant when item i starts on machines A and B, respectively. If item i-1 is operated on just before item i, then

$$t_i' = t'_{i-1} + A_{i-1}, \tag{8}$$

$$t_i = \max(t_{i-1} + B_{i-1}, t_i' + A_i, t_i' + D_i). \tag{9}$$

If by x_i we denote the idle time for machine B after it has finished item i-1 but before it starts to operate on item i, then

$$x_i = t_i - t_{i-1} - B_{i-1}. \tag{10}$$

It is easy to show (the proof is similar to that one of theorem 2) that one may restrict attention to programs of the type $A_q B_q$ since at least one optimal program must be of this type. Therefore, we are to find a n

element permutation p of numbers $1,\ldots,n$ which will minimize the total operation time $T(p)$ of program A_pB_p where

$$T(p) = \sum_{i\in p} B_i + \sum_{i\in p} x_i$$

Since

$$\sum_{i\in p} B_i = \sum_{i=1}^{n} B_i$$

is constant the problem reduces to one of finding a permutation p for which the total idle time of machine B

$$\sum_{i\in p} x_i = g(p)$$

will be minimal.

Divide the set $(1,\ldots,n)$ into two disjoint subsets s and s' where

$$s = \{(i) \mid A_i - B_i = E_i < 0\} \quad \text{and} \quad s' = \{(i) \mid E_i \geq 0\}.$$

Suppose s consists of ℓ elements while s' consists of $n-\ell$ elements. The following theorem holds.

<u>Theorem 5</u>. If the first ℓ elements of the permutation $p^* = (i_1, i_2,\ldots,i_\ell,i_{\ell+1},\ldots,i_n)$ belong to s while the remaining $n-\ell$ numbers are elements of s' and if the following relation holds

$$\max(A_{i_1},D_{i_1}) \leq \max(A_{i_2},D_{i_2}) \leq \ldots \leq \max(A_{i_\ell},D_{i_\ell})$$

as well as

$$\max(B_{i_{\ell+1}},D_{i_{\ell+1}} - E_{i_{\ell+1}}) \geq \max(B_{i_{\ell+2}},D_{i_{\ell+2}} - E_{i_{\ell+2}}) \geq \ldots$$

$$\geq \max(B_{i_n},D_{i_n} - E_{i_n})$$

then p^* is the optimal permutation.

Proof: Consider a permutation $(1,2,\ldots,n)$. One may assume $t_1' = 0$ and $t_1 = x_1$. Applying (8) and (10) we get

$$t_u' = \sum_{i=1}^{u-1} A_i, \quad t_u = \sum_{i=1}^{u} x_i + \sum_{i=1}^{u-1} B_i \quad \text{for } i = 1,2,\ldots,n$$

since: $t_i = x_i + t_{i-1} + B_{i-1}$, $t_1 = x_1 + 0$.

$$t_2 = x_2 + t_1 + B_1 = \sum_{i=1}^{2} x_i + B_1 \quad \text{and so on).}$$

Therefore

$$\sum_{i=1}^{u} x_i = t_u - \sum_{i=1}^{u-1} B_i \tag{11}$$

From (9) we find $t_u (t_u' = \sum_{i=1}^{u-1} A_i)$

$$t_u = \max\left(t_{u-1} + B_{u-1}, \sum_{i=1}^{u-1} A_i + A_u, \sum_{i=1}^{u-1} A_i + D_u\right).$$

Since

$$t_{u-1} = \sum_{i=1}^{u-1} x_i + \sum_{i=1}^{u-2} B_i,$$

therefore

$$t_u = \max\left(\sum_{i=1}^{u-1} x_i + \sum_{i=1}^{u-1} B_i, \sum_{i=1}^{u} A_i, \sum_{i=1}^{u-1} A_i + D_u\right).$$

Applying this result in (11) we have

$$\sum_{i=1}^{u} x_i = \max\left(\sum_{i=1}^{u-1} x_i + \sum_{i=1}^{u-1} B_i - \sum_{i=1}^{u-1} B_i, \sum_{i=1}^{u} A_i - \sum_{i=1}^{u-1} B_i, \sum_{i=1}^{u-1} A_i - \sum_{i=1}^{u-1} B_i + D_u\right) = X_u = \max\left(\sum_{i=1}^{u-1} x_i, \sum_{i=1}^{u-1} E_i + A_u, \sum_{i=1}^{u-1} E_i + D_u\right). \tag{12}$$

Formula (12) implies $X_1 = \max(0, 0+A_1, 0+D_1) = \max(A_1, D_1)$.

Similarly,

$$X_2 = \max[\max(A_1,D_1),\ (E_1+A_2),(E_1+D_2)] =$$

$$= \max[(A_1+D_1),\ (E_1+A_2),\ (E_1+D_2)] = \max_{1\le u\le 2}\left[\sum_{i=1}^{u-1} E_i+A_u,\ \sum_{i=1}^{u-1} E_i+D_u\right].$$

In general:

$$X_n = \max_{1\le u\le n}\left[\sum_{i=1}^{u-1} E_i+A_u,\ \sum_{i=1}^{u-1} E_i+D_u\right] = g(p). \qquad (13)$$

Consider an arbitrary permutation $p \neq p^*$. We will show that $g(p) \ge g(p^*)$. Without loss of generality we may assume that $p = (1,2,\ldots,n)$. Since $p \neq p^*$ it follows that p does not satisfy the assumptions of theorem 5. Therefore, there must exist in p two neighboring elements, say j and $j+1$, such that one of the following cases holds

(a) $j,j+1 \in s'$ and $\max(A_j,D_j) > \max(A_{j+1},D_{j+1})$

(b) $j,j+1 \in s'$ and $\max(B_j,D_j-E_j) < \max(B_{j+1},D_{j+1}-E_{j+1})$

(c) $j \in s'$, $j+1 \in s$

It will be shown that $p' = (1,2,\ldots,j-1,\ j+1,j,\ j+2,\ldots,n)$ satisfies condition $g(p') \le g(p)$. Let

$$P_u = \sum_{i=1}^{u-1} E_i+A_u, \quad \text{and} \quad Q_u = \sum_{i=1}^{u-1} E_i+D_u.$$

Then

$$g(p) = \max_{1\le u\le n}[P_u,Q_u], \quad \text{and} \quad g(p') = \max_{1\le u\le n}[P'_u,Q'_u],$$

where

$$P'_u = P_u, Q'_u = Q_u \quad \text{for} \quad u = 1,2,\ldots,j-1,j+2,\ldots,n$$

$$P'_j = \sum_{i=1}^{j-1} E_i+A_{j+1},\ Q'_j = \sum_{i=1}^{j-1} E_i+D_{j+1},\ P'_{j+1} = \sum_{i=1}^{j-1} E_i+E_{j+1}+A_j,$$

$$Q'_{j+1} = \sum_{i=1}^{j-1} E_i+E_{j+1}+D_j.$$

There are two cases: either $g(p) = \max(P_j, Q_j, P_{j+1}, Q_{j+1})$ or $g(p) \neq \max(P_j,Q_j,P_{j+1},Q_{j+1})$.

In the first case $g(p) = g(p')$ which implies $g(p') \leq g(p)$.

Consider the second case. We are to show that

$$\max (P'_j, Q'_j, P'_{j+1}, Q'_{j+1}) \leq \max(P_j, Q_j, P_{j+1}, Q_{j+1}),$$

i.e. that the following inequality holds

$$\max \left[\sum_{i=1}^{j-1} E_i+A_{j+1}, \sum_{i=1}^{j-1} E_i + D_{j+1}, \sum_{i=1}^{j-1} E_i+E_{j+1}+A_j, \sum_{i=1}^{j-1} E_i+E_{j+1}+D_j \right] \leq$$

$$\leq \max \left[\sum_{i=1}^{j-1} E_i+A_j, \sum_{i=1}^{j-1} E_i+D_j, \sum_{i=1}^{j} E_i+E_j+A_{j+1}, \sum_{i=1}^{j} E_i+E_j+D_{j+1} \right]$$

Subtracting $\sum_{i=1}^{j-1} E_i$ from both sides of the last inequality we get

$$\max(A_{j+1},D_{j+1},E_{j+1} + A_j,E_{j+1} + D_j) \leq \max(A_j,D_j,E_j+A_{j+1},E_j+D_{j+1}). \quad (14)$$

Denote by $\bar{L}$ and $\bar{R}$ the left and right hand side of (14)

Consider case (a)

According to the assumption in the theorem $E_j < 0$ and $E_{j+1}<0$. Therefore

$$A_{j+1} > E_j+A_{j+1}, D_{j+1} > E_j+D_{j+1}, A_j > E_{j+1}+A_j \text{ and } D_j > E_{j+1}+D_j.$$

Inequality $\bar{L} < \bar{R}$ will be true if

$$\max(A_{j+1},D_{j+1}) < \max(A_j,D_j). \quad (15)$$

But (15) holds in view of the assumption of the theorem. Therefore $\bar{L} < \bar{R}$ which implies $g(p') \leq g(p)$.

Consider case (b).

It is assumed $E_j \geq 0$ and $E_{j+1} \geq 0$. This implies

$$A_j \leq E_{j+1}+A_j, D_j \leq D_j+E_{j+1}, D_{j+1} \leq E_j+D_{j+1} \text{ and } A_{j+1} \leq E_j+A_{j+1}.$$

Relation $L < R$ holds if

$$\max(E_{j+1}+A_j,E_{j+1} + D_j) < \max(E_j+A_{j+1},E_j+D_{j+1}).$$

Subtracting $E_j + E_{j+1}$ from both sides we get

$$\max(A_j-E_j,D_j-E_j) < \max(A_{j+1}-E_{j+1},D_{j+1}-E_{j+1}).$$

This may be written in the form

$$\max(B_j, D_j - E_j) < \max(B_{j+1}, D_{j+1} - E_{j+1}). \qquad (16)$$

Since (16) holds, by assumption, it follows that $\bar{L} < \bar{R}$ and $g(p') < g(p)$.

Consider case (c).

By assumption $E_j \geq 0$ and $E_{j+1} < 0$ which leads to

$$A_{j+1} \leq E_j + A_{j+1} \;,\; D_{j+1} \leq E_j + D_{j+1} \;,\; A_j + E_{j+1} < A_j \;,\; D_j + E_{j+1} < D_j.$$

This in turn implies $\bar{L} \leq \bar{R}$, $g(p') \leq g(p)$. Thus we have showed that if permutation p satisfies one of the conditions (a), (b), (c) then there exists a permutation p' which emerges from p by transposing two neighboring elements of p and where $g(p') \leq g(p)$. This implies that for any permutation $p \neq p^*$ there exist at least a finite sequence of permutations $p, p', p'', \ldots, p^*$ with $g(p) \geq g(p') \geq g(p'') \geq \ldots \geq g(p^*)$ where each permutation of the sequence is constructed from the preceding one by a transposition of two elements. Permutation $g(p^*)$ is optimal since for any permutation p, $g(p) \geq g(p^*)$.

Example: Consider a 2x5 problem with the following data (Table 7)

	A_i	B_i	D_i	E_i	$D_i - E_i$	$\max(A_i, D_i)$	$\max(B_i, D_i - E_i)$
1	3	2	3	1	2		2
2	2	4	2	-2	4	2	
3	1	5	4	-4	8	4	
4	4	3	2	1	1		3
5	3	4	1	-1	2	3	

Table 7

Here $s = (2,3,5)$ and $s' = (1,4)$. According to theorem 5 in the optimal sequence: 1) elements 2,3,5 appear before elements 1,4. 2) the numbers 2,3,5 are arranged so that values $\max(A_i,D_i)$ form a nondecreasing sequence, 3) the numbers 1,4 are arranged in such a way that $\max(B_i,D_i-E_i)$ form a nonincreasing sequence.

Therefore, permutation $q = (2,5,3,4,1)$ is optimal. Figure 13 shows the working schedule of program A_qB_q.

Figure 13

Remark: If $D_i \leq A_i$ for each $i=1,\ldots,n$ then as mentioned, the problem reduces to a classical Bellman-Johnson case. Then according to (9) t_i becomes $t_i = \max(t_{i-1}+B_{i-1}, t'_i+A_i)$. For this case the inequalities appearing in the statement of theorem 5 will be as follows

$$A_{i_1} \leq A_{i_2} \leq \cdots \leq A_{i_\ell}, \quad B_{i_{\ell+1}} \leq B_{i_{\ell+2}} \leq \cdots \leq B_{i_n},$$

since

$$\max(A_t,D_t) = A_t \text{ and } \max(B_t,D_t-E_t) = \max(B_t,D_t-A_t+B_t) = B_t.$$

This implies a rule of constructing an optimal sequence, which rule was stated in Section 2.

7 Some Properties of the Approximate Solution Method of the mxn Case.

In Section 3 an approximate method of solving the mxn case was given. We will show that this method is an exact method for the problems presented in Section 2.

Consider the 2xn Bellman-Johnson Case. According to the method mentioned, one must first solve $\binom{n}{2}$ different 2x2 problems for each

$i,j = 1,\ldots,n$, $i \neq j$ by the method given in Section 3. Consider one of the 2x2 cases (see Figure 14).

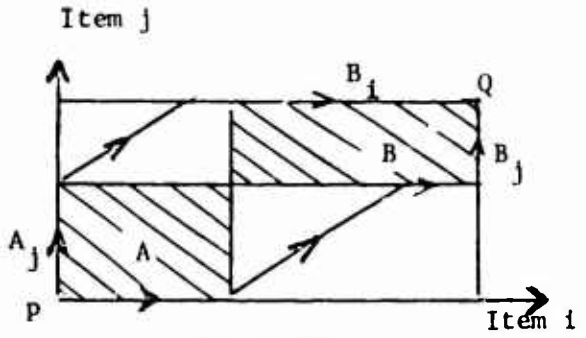


Figure 14

It is easy to see that the optimal line can only be either one of the two (indicated on Figure 14 by arrows) linking P and Q. The lower line corresponds to the program $A_{(i,j)}\ B_{(i,j)}$ while the upper line corresponds to the program $A_{(j,i)}\ B_{(j,i)}$.

By z and z' let us denote the total length of all vertical segments corresponding to the lower and upper line (this is the total waiting time for item i in the 2x2 problem) Then (see Figure 14)

$$z = B_j + \max\ (0,\ A_j - B_i),$$

$$z' = A_j + \max\ (0,\ B_j - A_i).$$

The sufficient condition for the lower line to be optimal is

$$B_j + \max(0,\ A_j - B_i) < A_j + \max(0,\ B_j - A_i).$$

where an equality sign may also appear. This inequality may be rewritten as

$$\max(B_j,\ A_j + B_j - B_i) < \max\ (A_j,\ A_j + B_j - A_i).$$

Subtracting $A_j + B_j$ from both sides we get

$$\max(-A_j,\ -B_i) < \max(-B_j,\ -A_i).$$

which after simple transformations becomes the equivalent form

$$\min(A_j,\ B_i) > \min(A_i,\ B_j). \qquad (17)$$

Condition (17) is identical with that of Bellman-Johnson (see [7]). Since (17) is transitive we will get from the $\binom{n}{2}$ problems of the type 2x2 a solution of the 2xn problem identical with the optimal solution obtained by Johnson's method.

Remark: One may derive condition (17) directly by looking for a line for which the total length of the 45° segments is maximal. Let us denote such a length for the lower and upper line by $\bar{z}$ and $\bar{z}'$ respectively. Then (see Fig. 14)

$$\bar{z} = \min(A_j, B_i), \qquad \bar{z}' = \min(A_i, B_j).$$

The lower line represents an optimal program if

$$\min(A_j, B_i) \geq \min(A_i, B_j),$$

which is identical with condition (17).

Consider a 3xn Bellman-Johnson case while additionally assuming that for each $i,j = 1,2,\ldots,n$, at least one of the relations $B_i \leq A_j$, $B_i \leq C_j$ holds.

Here we consider $\binom{n}{3}$ different 3 x 2 problems. Take one of those for items i and j, say, (see Figure 15). According to Johnson in [7] we may look for an optimal program among programs of the type $A_p B_p C_p$. This implies that one can find the optimal line by examining only two lines as presented on Figure 15.

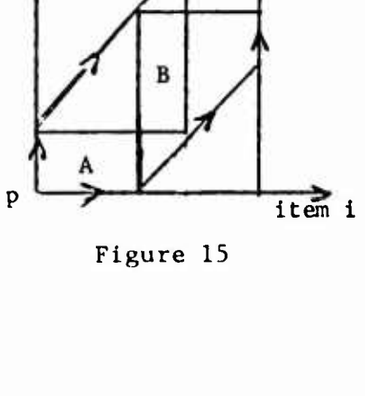


Figure 15

Let us denote by $\bar{w}$ and $\bar{w}'$ the total length of the 45^{o} segments of the lower and upper line respectively. Then (see Figure 15)

$$\bar{w} = \min(B_i + C_i, A_j + B_j),$$

$$\bar{w} = \min(A_i + B_i, B_j + C_j).$$

So if $\bar{w} > \bar{w}'$

$$\min(B_i + C_i, A_j + B_j) > \min(A_i + B_i, B_j + C_j), \qquad (18)$$

then the lower line is optimal and this means that program $A_{(i,j)}$ $B_{(i,j)}$ $C_{(i,j)}$ is optimal. Condition (18) is identical with that given by Johnson in [7].

The transitivity of (18) implies that the mxn method always solves the 3xn Bellman-Johnson case under the condition $B_i \leq A_j$ or $B_i \leq C_j$. One can easily show that this method solves the 3xn Bellman-Johnson case when $B_i \geq \max(A_j, C_j)$ for each i and j as well as the 2xn case from Chapter 4.

References

[1] Akers, S. B., J. Friedman, "A Non-numerical Approach to Production Scheduling Problems," Operations Research, v.3, pp. 429-442 (1955)

[2] Akers, S. B., "A Graphical Approach to Production Scheduling Problems," Operations Research, v.4, pp. 244-245 (1956).

[3] Bellman, R., "Mathematical Aspects of Scheduling Theory," Journal of the Society for Industrial and Applied Mathematics, v. 4, pp. 168-205 (1956).

[4] Bowman, E. W., "The Schedule-Sequencing Problem," Operations Research v. 7, pp. 621-624 (1959).

[5] Dantzig, G. B., "A Machine-Job Scheduling Model," Management Science, v. 6, pp. 191-196 (1960).

[6] Gomory, R, E., "Outline of an Algorithm for Integer Solutions to Linear Programs," Bulletin of the American Mathematical Society, v. 64, pp. 275-278 (1958).

[7] Johnson, S. M., "Optimal Two and Three Stage Production Schedules with Set-up Times Included," Naval Research Logistics Quarterly, v. 1, pp. 61-68 (1954).

[8] Mitten, L. G., "Sequencing n Jobs of Two Machines with Arbitrary Time Lags," Management Science, v. 5, pp. 293-298 (1959).

[9] Szwarc, W., "Solution of the Akers-Friedman Scheduling Problem," Operations Research, v. 8, pp. 782-788 (1960).

[10] Wagner, H. M., "An Integer Linear Programming Model for Machine Scheduling," Naval Research Logistics Quarterly, v. 6, pp. 131-140 (1959).

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)
Graduate School of Industrial Administration
Carnegie Institute of Technology

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP
Not Applicable

3. REPORT TITLE

ON SOME SEQUENCING PROBLEMS

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
Technical Report, July, 1967

5. AUTHOR(S) (Last name, first name, initial)

SZWARC, Wlodzimierz

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
July, 1967	40	10

8a. CONTRACT OR GRANT NO.
NONR 760(24)

b. PROJECT NO.
NR 047-048

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)
Number 103
Management Sciences Research Report

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
none

10 AVAILABILITY/LIMITATION NOTICES

Distribution of this document is unlimited

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
none	none

13. ABSTRACT

The (mxn) sequencing problem may be characterized as follows: There are m machines which can produce a piece consisting of n parts. Each part has a determined order in which it is processed through the machines. It is assumed that each machine cannot deal with more than one part at a time and that the processing required for each part can be accomplished only on one machine. That is, the machines are all specialized so that alternate machines for the same processing on a part is not possible. The problem is to find the best production plan consisting in sequencing the different parts so as to make the whole amount of time from the beginning of work till the piece is completed the shortest possible. Such a plan is called an optimum one. In the first 4 sections of this paper, the problem (2xn) is solved for the (2xn) case in which the order in which parts come on the machine is not constrained by futher assumptions. The remainder of the paper then takes up: i, the (3xn) problem of Bellman-Johnson (viz. the technological processing order through the machine is the same for all parts), for several new special cases; ii, the 2xn problem of sequencing when delay times must also be considered; and, iii, some properties of an approximating method for solving (mxn) problems, including a delineation of cases when the approximating method will yield optimal solutions.

DD FORM 1 JAN 64 1473

Unclassified

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Sequencing Production Scheduling Machine Loading Job Shop Scheduling Linear Programming Integer Programming Graph Theory						

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2a. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2b. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year; or month, year. If more than one date appears on the report, use date of publication.

7a. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8a. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8b, 8c, & 8d. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9a. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9b. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through

_____."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through

_____."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through

_____."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS)*, *(S)*, *(C)*, or *(U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, roles, and weights is optional.